

## A LEVEL

*Exemplar Candidate Work*

# COMPUTER SCIENCE

**H446**

For first teaching in 2015

**H446/03 Summer 2017  
examination series**

**Set A – Low**

Version 1

# Contents

<b>Introduction</b>	3
Exemplar 3	4
Commentary	31

# Introduction

These exemplar answers have been chosen from the summer 2017 examination series.

OCR is open to a wide variety of approaches and all answers are considered on their merits. These exemplars, therefore, should not be seen as the only way to answer questions but do illustrate how the mark scheme has been applied.

Please always refer to the specification (<http://www.ocr.org.uk/Images/170844-specification-accredited-a-level-gce-computer-science-h446.pdf>) for full details of the assessment for this qualification. These exemplar answers should also be read in conjunction with the sample assessment materials and the June 2017 Examiners' Report to Centres available on the OCR website <http://www.ocr.org.uk/qualifications/>.

The question paper, mark scheme and any resource booklet(s) will be available on the OCR website from summer 2018. Until then, they are available on OCR Interchange (school exams officers will have a login for this).

It is important to note that approaches to question setting and marking will remain consistent. At the same time OCR reviews all its qualifications annually and may make small adjustments to improve the performance of its assessments. We will let you know of any substantive changes.

# Exemplar 3 – Set A (Low)

## Programming project (non exam assessment)

Learners will be expected to analyse, design, develop, test, evaluate and document a program written in a suitable programming language.

**GEOGRAPHY REVISION**

**QUIZ PROJECT**

**H446-03**

## Analysis

### Problem Identification

The problem is that geography students revise from booklets, not answering questions. Answering questions is an effective way to revise. The purpose of the project is to help those students revise by giving them different questions for them to answer. The program will display a question and wait for the user's input to answer the question. The answer is then checked. Every question will be random and from a different topic, if they answer the question correctly, they will be given another question from another topic. However, if they answer the question incorrectly, they will automatically be given a random question from the same topic. They will then have to answer 3 questions correctly to move on to a different topic. Also the questions should not repeat themselves during each session. Their score will be recorded and stored. They will be able to see their score and answers or previous results to see their progress. This may also be displayed on a graph.

The user will not be able to choose the questions as they will be selected automatically by the program, tracking their progress. There are 5 different topics in A-level geography with a lot of content. Therefore, this means that the program will be more useful to students than revising from the booklets as instead of reading through the booklet, they can answer questions from all topics and find the areas where they have to spend more time on.

The program will have to contain an algorithm to decide which questions to display to the user, a checking/marking system so that the student will be able to see which questions they got wrong, storage algorithm so that all results to the questions are stored and can be again used in the program. Also program will have to contain a lot of questions from each topics and from different areas of the topic and an algorithm to change the questions, depending on user's input (if the question is correct or not). Furthermore, the program should add up all the scores and display the average score/grade of the student at which they are currently performing.

My project should be better for the user rather than using paper notes as the program tests the user's knowledge and helps them to learn geography by providing the right answers when a question is answered incorrectly. It is difficult to test yourself using paper notes by yourself and then the person would have to mark it themselves as well.

I believe that by using a computer, the program will be much more useful for the user than the paper based revision booklets. By using the program on a computer, the user will only have to answer questions and their score will always be saved in the same place. The program would mark the answers of the user and correct them if needed, whereas when using a revision booklet, or answering questions on paper, the user would have to mark the answers themselves, which means that human error is possible. Also, marking the answers is time consuming & this time could be used to answer more questions. The program however, would mark the questions for the user and display the right answer if the user entered an incorrect answer.

The program is possible to make on a computer, as there are multiple ways in which the final solution can be achieved. For example, to save the score or load it, writing-to-file and reading-from-file allows this. This also means that registration and login is possible, as data is stored permanently in files, unless deleted by the user.

My project will be solved using multiple computational methods. For example, abstraction will be necessary for this project in order to make it as useful and user-friendly as possible. By using abstraction, I will change or delete unnecessary sections of the program. It will also make the code easier to understand by other people as there will be less unnecessary variables. Abstraction will also allow me to break down the program and create a flow chart for my design so that it will highlight what features the program must have. I will have to think ahead as well so that I know what variables I will use in the program. It will also be possible to break down the program into sub-sections as there will be multiple parts of the program that needs to be done. For example, login section, questions section, results...

Computational methods will be most useful in the "Questions" section as it will contain multiple functions such as writing-to-file, reading-from-file, checking answers, randomising questions... Abstraction for example will make this section easier to program & programming it will be more efficient as due to using abstraction and thinking ahead, I will avoid spending time on writing unnecessary code which would be deleted afterwards. Without using a computer, this project would not be possible as the program checks the user's answer and stores the score... Therefore, without using a computer, the user would have to check the answer themselves and write the score on paper. This would be less efficient than on a computer as the program does these things automatically without user input.

#### Users

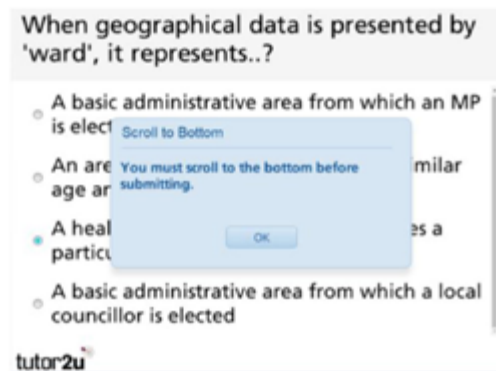
This project is aimed at A-Level students who study geography.

The program's objective is to improve the students' knowledge of geography (A-Level) by giving them randomised questions from different topics and help the students to find the areas in A-Level geography which they need to improve on. User's requirements will be met if the program will be more effective way of revising than reading from booklets and if the program will be simple, easy to use and helping the students to point out the topics or areas in topics on which they need to work on.

#### Research

There are a few programs that are similar to my project. However, those programs display the answer at the end, whereas I would like the answer to be shown right after the user enters the wrong answer. Also, most of the programs I have seen, work as a browser application and so when browser history & data is removed, so are the settings and the rest of the temporary data. Some however, store user data in their server, however as I do not have a server, the program will be offline. Furthermore, overall there aren't many programs specifically for A-level geography.

One of the programs I researched was "Tutor2u" (geography section). It provides the user with 10 multi-choice style questions. The correct answers and the score is displayed at the end. The questions are randomised but there are some bugs, e.g. false error message saying to scroll down to the bottom of the screen when I already did that:



It's a lot different to what I would like my program to do as this program only has multi-choice questions. There are few features I would like my program to have, also used in this program. For example, I want the user's answers to be marked and corrected & the questions to be random. I also may include in my program a multi-choice style answers as it may help the user to remember the correct structure of the answer to a specific question. Also, it may take less time for the user to select the answer instead of writing it out and it the marking system would be more accurate as spelling mistakes would no longer take place or any other human errors.

### Essential Features

One of the most important features is the algorithm to randomise the questions and choose the category based on user's input. Another essential feature is the option for the user to see their past results and compare them to the recent results to see their progress, so they can improve it in the future. These are main reasons why this project is most suitable to be solved by computational methods as it would be impossible to present the user with random questions otherwise. Also, the algorithms will be doing most of the work, reducing the need for user input and as the answers will be marked for the user and the correct answers will be displayed, it means that the user would save a lot of time as everything is done automatically.

The correcting system is essential, as all users' answers have to be corrected as accurate as possible and then display whether it's correct or not. The answers have to be stored as well so that the user can keep track of their progress/results. The program should have a login system for security and privacy of the user as well.



Moreover, the program should have an option to display the results on a graph so that the user can quickly see their progress very clearly. This will make the program more user-friendly and more efficient.

My project will have multiple files storing data, which means that data will be stored permanently. Data that will be stored permanently includes login information and user's score. This means that the program will have to contain write-to-file and read-from-file algorithms.

In the "Questions" section the user will be presented with a random question and the user will be required to input an answer & use a button to proceed.

Also, the user will be able to see all their results in a table, so it's clear.

#### Limitations

One of the limitations is time as setting up all questions with the answers is very time consuming and therefore, the less time I have, the less questions I will be able to make. This will also affect the quality of the marking system as making it accurate is time consuming as well as it requires multiple algorithms as it can be affected by spelling mistakes, spaces and other factors. My coding skills are also a limitation as it will take me longer to complete certain complex tasks. Visual Basic has 2 limitations;

*'A single project can contain up to 32,000 "identifiers" (any non-reserved keyword), which include, but are not limited to, forms, controls, modules, variables, constants, procedures, functions, and objects. Note that the actual number of identifiers is limited to available memory.'*

*'Variable names in Visual Basic can be no longer than 255 characters, and the names of forms, controls, modules, and classes cannot be longer than 40 characters. Visual Basic imposes no limit on the actual number of distinct objects in a project.'* - Microsoft

#### Solution Requirements

The final solution will ask the user for their login details, which will take them to the menu. Then the user will have to choose if they want to start answering questions, see their results or their progress. The program will have to randomise all the questions and display a question depending on the user's answer for the past question. Also, the questions will have to be marked by the program and/ or display the answers.

The hardware requirements are a mouse, keyboard, monitor and a desktop or a laptop. This program will not work on mobile phones, tablets or any other devices. The only software requirement is the operating system: Windows.

I have chosen those requirements as most students do not use mac computers and mobiles phones have smaller screens, it's harder to type in the answer and could be too distractive.



### System Requirements

- Pentium 90MHz or higher microprocessor.
- VGA 640x480 or higher-resolution screen supported by Microsoft Windows.
- Microsoft Windows NT 4.0 or later, or Microsoft Windows 95 or later.
- 24 MB RAM for Windows 95/98, 32 MB for Windows NT.

These requirements had been obtained from a Microsoft website, based on running visual basic programs. The disk space required will depend on the size of the final version for the program. However, CPU usage will be affected by different functions in the program, its size and multiple other factors. This will affect the amount of RAM used, which might mean that Pentium 90MHz or similar micro-processors may not have enough power to run my program smoothly.

### Success Criteria

The project will be a success once it is working without any errors as planned and is more effective for students for revision than going over the geography booklets. The program should help the students to improve their knowledge of geography. Also, the marking system must work fairly accurately.

To measure success criteria, I will have to test the marking system and if the questions are correctly linked to the answers. The marking system will have to take spelling mistakes into account. The program has to be user-friendly, it must be efficient and must meet the user's needs, which means that the program must be helpful and effective to use for revision. The most important areas that need to be included in the program are: marking system so the user knows if they answered incorrectly and so they know the correct answer & generator of random questions has to work correctly and match the right answers.

**Comment [JM1]:** Measurable  
HOW?

**Design**

## Visual Design

Starting screen

A rectangular window with a light blue background. At the top is a white rectangular box labeled "LOGO". Below the logo, three blue rectangular buttons are stacked vertically, labeled "Login Button", "Register Button", and "Quit Button".

Log-in Window

A rectangular window with a light blue background. At the top is a white rectangular box labeled "LOGO". Below the logo, there are two columns of input fields. The left column has labels "Username:" and "Password:" next to white input boxes. The right column has white input boxes labeled "Username" and "Password". Below the input fields, there are two blue rectangular buttons: "Login button" on the left and "Back" on the right.

Register Window

A rectangular window with a light blue background. At the top is a white rectangular box labeled "LOGO". Below the logo, there are two columns of input fields. The left column has labels "Username:", "Password:", and "Admin code" next to white input boxes. The right column has white input boxes labeled "Username", "Password", and "Required to make new account". Below the input fields, there is a single blue rectangular button labeled "Back" on the right side.

**Comment [JM2]:** System map is required  
Show the breakdown of the system?

Quit Window

A diagram of a 'Quit Window' with a light blue background. At the top is a white rectangular box containing the text 'LOGO'. Below this box is the question 'Are you sure you want to quit?' in black text. Underneath the question are two blue rectangular buttons: the top one is labeled 'Yes' and the bottom one is labeled 'No'.

Menu Window

A diagram of a 'Menu Window' with a light blue background. At the top is a white rectangular box containing the text 'LOGO'. On the left side, there is a vertical list of blue rectangular buttons: 'Start Questions', 'My Recent Results', 'My Progress', 'Past Results', 'Quit', and 'Help'. On the right side, there are three blue rectangular buttons stacked vertically: 'Username', 'Currently working at:', and 'My recent result'.

### Questions

LOGO	
Start Questions	Question number
My Recent Results	QUESTION
My Progress	Answer Box
Past Results	
Quit	
Help	

### My recent results

LOGO	
Start Questions	
My Recent Results	Result Dates
My Progress	
Past Results	
Quit	
Help	

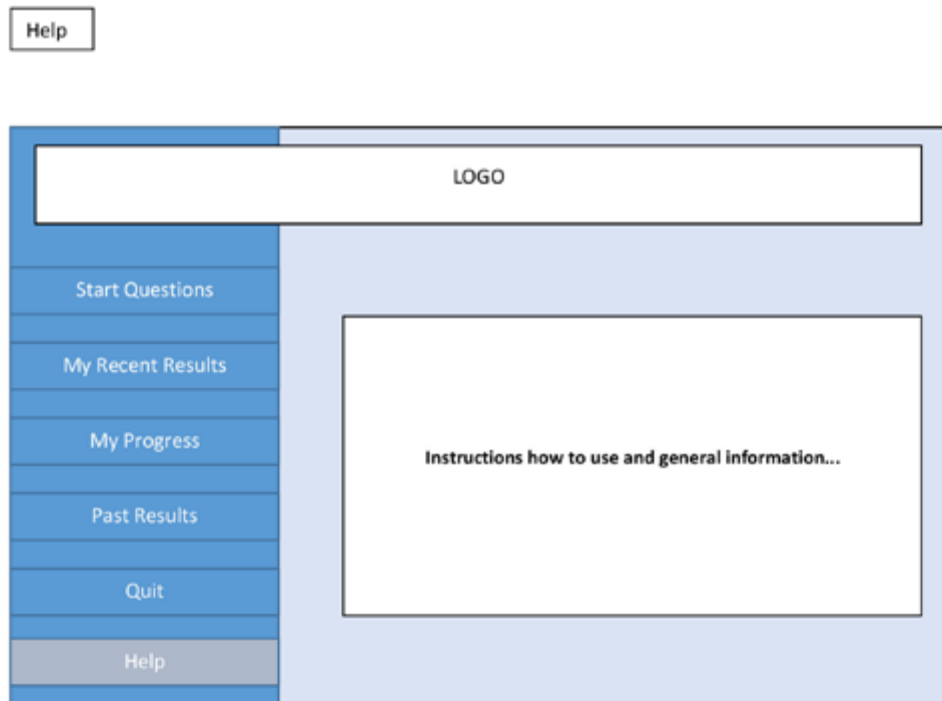
My progress



Past Results

The screenshot shows the same web application interface as above, but with the "Past Results" menu item highlighted in the sidebar. The main content area now displays a table with two columns: "Result" and "Dates". The table has 10 rows, with the first row containing the column headers and the subsequent 9 rows containing the word "Result" in the first column and empty cells in the second column.

Result	Dates
Result	
Result	
Result	
Result	
Result	
Result	
Result	
Result	
Result	
Result	



### Main

I will leave the layout of the program as it is shown above however I will change the colours and the background to give it a more professional look. I made the navigation bar on the left so that it will be always visible to the user, making it more user-friendly as it will make it very easy to navigate around the program. Any logos or pictures will be related to A-level geography.

**Start Questions** – This section will have to perform multiple functions. For example, it will have to read-from-file in order to display the questions. The questions will have to be randomised before they are displayed to the user. The algorithm will also have to contain a marking system so that user's answer is checked. The algorithm then would continue but the next step would rely on if user's answer is correct or not.

**Login** – When a user registers, their details will be safely stored somewhere in the computer. Therefore, when the user tries to login, their input is compared to their original login information, then the program decides whether it's correct or incorrect, which will determine if they can enter the system. Admin code will stop unauthorised access to the program as to register, the user will have to be given the admin code, otherwise it will be impossible for them to register. This is done so that only students can see each other's results and to protect personal information (student's names).

**My recent results** – My recent result page will show the results from the most recent questions they answered.

**My progress** – 'My progress' will display a graph/chart which will show all the user's results from the past, which will overall show time over results. Therefore, the students can see if they improve in geography, stayed at the same level or if they need to improve.

**Quit** – When quit button is clicked, to prevent data loss, the program will display the message if the user really wants to quit so that they won't lose their recent test results. Also, every time the program is closed, the program will automatically save all data.

**Help** – Help section will give instructions, if the user needs any, how to use the program or some solutions if any error occurs. If something is unclear for them, e.g. what a section of the program is for, they will be able to find their answer in the help section.

All of the sections mentioned above, will be put together to make a single fully working solution. Any useless features should be deleted so that the program is more efficient and user-friendly.

#### Plan

In order to store the data, an algorithm will be required which will sort the data and link them together correctly. To do this, I may use an array to store information about one person in a single file instead of having multiple files linked together. It's possible to link multiple files together, matching the data together. However this method will be more likely to make an error or even crash.

Also, I may use both methods as different data will have to be saved, e.g. questions with answers, information about the user and the progress of the user.

Furthermore, if the user will want to find an answered question or a topic, a search option will be required. For this, I will not be using a binary search as the questions will be generated randomly, therefore the list of data will always be changing and it will be unordered. The list will most likely not be long as well which means that other search methods will fit this purpose better. For example, serial search.

To prevent some errors or crashes, all sections requiring the user to input text, will be limited to certain characters so that it's always valid. E.g. Letters, signs and numbers for password but only text and numbers for the username so that it's saved properly and may be found using the search algorithm.

#### How the program will function

Firstly, once the user logs in and is at the stage where they selected to answer questions, there will be an algorithm which will choose a random question from a random topic. Then the user has to answer the question in the space provided. Once the user enters the answer, the answer is compared to the correct answer, if it's correct the user will be given another randomly generated question from a random topic. However, if the question was incorrect then the user will be given a question from a similar or the same topic. This will happen until the user gets one to three questions correct from that topic. This is done so that the user knows what topic they need to improve at in more detail. Each question will have its own ID so that the questions are not repeated, therefore the end result of the user is not affected by repeated questions.

Also, there will be an algorithm which will deal with spelling mistakes. For example, if the user makes a spelling mistake, the word/answer will be displayed, highlighting the mistakes. This is done as incorrect spellings will cause the user to lose marks on a geography test. If the spelling is correct however this process will be skipped.

Once the user stops to answer questions, either due to reaching the end of the test or ending before it has ended, the score, along with the user's name/username is recorded and stored in a secure file.



This is so that the data can be used later on, so that the user can see their progress and to see the most recent result as user's progress is a very important part in the program's success criteria, as this will show how useful the program is.

### User Features

My goal is to make the program as much user friendly as possible so that it's more efficient for the user as it won't require a lot of time to learn how to use the program.

The program will have the navigation bar on the left at all times therefore it doesn't matter where the user decides to go as the navigation bar will always be on the left so it's visible at all times, which means that it will be easier to get around. Also, the background and all colours will not be bright so that the program can be used for longer periods of time without hurting the user's eyes.

Auto marking system is also a user-friendly feature as the user's answers to the questions will be marked or corrected automatically without requiring any more input from the user, except the answer. This is a useful feature as it will also save a lot of user's time.

Furthermore, the program will be tested multiple times in different ways to make sure that the program is reliable and there are no errors. However, if a crash does occur, there will be a backup in case any data is lost.

The user will be able to navigate around the program by using buttons as shown on the plan above. There will also be a results table to display the results so that the user can see their progress. The results which will be displayed in that table will have to be stored in a file, so that data can be stored permanently rather than temporarily as the program is running.

### Data Structures

For storing questions and user's answers, a database will be used so that it's possible to link the data back to the program, display it on a graph and so that all data is in a correct category and linked together correctly. If all data would be stored as single, separated text files, it would be extremely difficult to link data correctly together, e.g. question with the correct answer. This also makes it more likely to cause a lot of errors, crashes or just display incorrect data. A 2D array may also be used to store user's details correctly, without having to split user's information into multiple separate documents and then putting them back together.

A second database will also be made to store questions and answers. Having the second database will be simpler than storing everything in one database and therefore, it will reduce the chance of making a mistake.

Furthermore, a third database will be password protected or encrypted, for storing user's information/data like username, password and name. This information will be store in a separate database to increase the security and reduce the chance of data theft and unauthorised access.

The main algorithm will randomise the questions for the user and matching it with the correct answer, comparing it with the user's answer. The second algorithm will store certain data in a database. There will also be an algorithm for the login section, separated from others.

The algorithm that will randomise the questions will have to be in a conditional loop. E.g. Loop randomised questions from random category 10 times, if user input does not match with the correct

answer, loop random questions from that category or similar until user input matches with the correct answer.

Most variables will be declared locally, so that they won't be affected by other sections of the programs and therefore decreasing the chance of logical error or a crash. For example, Login, Password, Question, Question number, Answer, Result variables will be declared locally. There will be some however which will be declared globally so that they don't have to be re-written in each section of the program. For example, the randomised integer variable will be stored as a global variable so that it may be used in questions & answers section of the program.

Progress section will allow the user to view previously obtained scores so that they can see if they improved. The scores will be recorded in a score table in the program. The score will be obtained from a csv file.

CSV files will be used to store data. These files will be used by the program by the "write-to-file" and "read-from-file" functions. The files will be the program's permanent storage.

Also, some data from specific CSV files may be loaded into record structure so that it is easier to work with the data in the files. This will prevent many possible logical errors. This will also allow me to overwrite, delete or modify the data accurately in the files.

### Testing

Program functionality will have to be tested to make sure that it meets success criteria and is user-friendly. I will test the program using different forms of data as an input to check for multiple potential errors. For example, I will use different characters for the registration and login section to see if a crash will occur. Also, each part of the code will have to be tested (each function) to make sure there are no logic errors, which would cause the program to function incorrectly. This is because when logic error occurs, the program will still run but the outcome will be incorrect.

Also, I will have to make sure that all functions of the program function correctly and I will be improving them so that they are more useful and make the program more user-friendly.

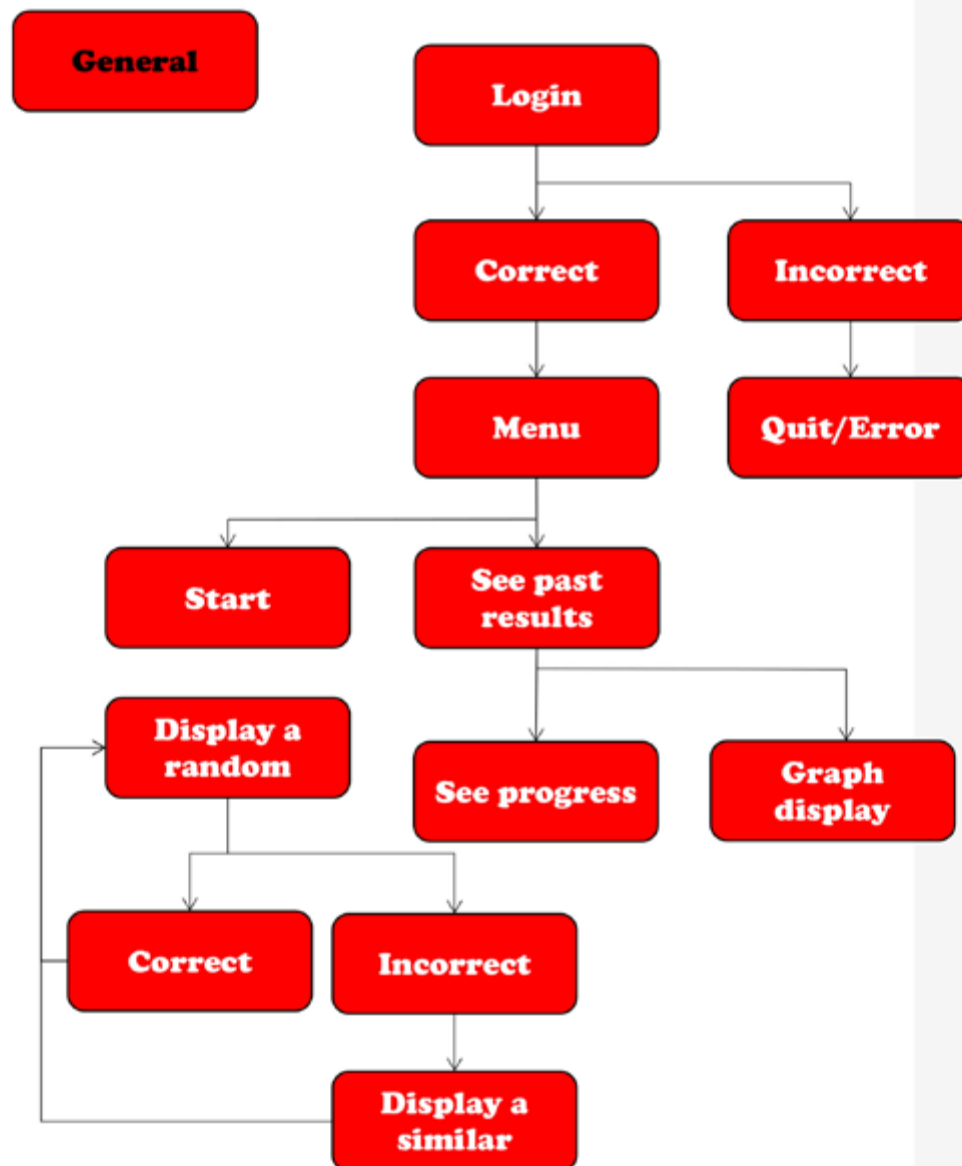
For the login section, I will have to test if the program matches the user input to the correct username and password. Also, if there are any ways where the user can log into the program without having to register first.

For registration section, I will have to make sure that the write-to-file code functions correctly and so that the user can't overwrite an already made account. The login and registration sections have to be tested to make sure that the security is decent in the program. Therefore, other users will not be able to log into another user's account without knowing their username and password.

For the questions section, there will be multiple areas that will have to be tested. For example, I will need to check if the questions are generated correctly, if the write-to-file and read-from-file functions work correctly, as well as randomising algorithm. The variables should be declared correctly and the record structure should be loaded up correctly. This section is the most important and has to be tested carefully as it determines the program's usefulness and efficiency.

Once the program is fully operational, user-friendly and efficient then I believe that the program will meet the success criteria. The sections described above, will determine if the success criteria has been met. However, usability features will have to be tested as well to make sure that the program looks professional and the writing is clear.

Flow chart of the System



### Code Plan

#### Registration

Initialise **Username** as string = UsernameTextbox.text

Initialise **Password** as string = PasswordTextbox.text

Fileopen(1, "Location/Login.csv", openmode.output)

If **Username** exists then

Msgbox("Username already in use!")

Else

Msgbox("Registration successful!")

Printline(1, **Username** & "," & **Password**)

Application.restart()

End if

Fileclose(1)

-Registration allows the user to create a new account which will be used mainly to save scores. This will allow the user to view these scores and see if there has been any improvement.

#### Login

Initialise **Search** as boolean = false

Fileopen(1, "Location/Login.csv", openmode.input)

Do

If lineinput(1).Contains(Loginusernametextbox.text) and (LoginPasswordtextbox.text) then

Msgbox("Login Successful!")

**Search** = true

Me.hide

Menu.show

Else

**Search** = false

Loop Until (EOF) or **Search** = true

Msgbox("Username not found")

End if

-Login section allows the user to login to their account.

### Menu

If Start is pressed then

Me.hide

Start.show

Elseif Progress is pressed then

Me.hide

Progress.show

Elseif Help is pressed then

Me.hide

Help.show

Elseif Log out is pressed then

Application.restart

Elseif Quit is pressed then

Application.exit()

End if

-Menu section will allow the user to navigate around the program. Once a new page is opened, the current page will not close, instead it will be hidden as there are no active processes active on that page so it will not affect the cpu much.

### Questions

Fileopen(2, "Location/Question.csv", openmode.input)

Initialise **random** as integer

If Next button is pressed then

Randomize()

Random = Cint(Int((((**number of questions**) \* Rnd()) + 1))

Initialise **Questions** as string

Do

**Question** = Lineinput(2)

Loop = (Random)

QuestionBox.text = **Question**

Fileclose(2)

If Submit answer is pressed then

```

Fileopen(3, "Location/Answers.csv", openmode.output)
If Lineinput(3, random) = Answerbox.text then
Msgbox("Correct!")
Scorelbl.text = Scorelbl.text + 1

```

```

Else
Msgbox("Incorrect")
Totallbl.text = Totallbl.text + 1
End if
If Back is pressed then
Fileopen(4, "Location/Score.csv", openmode.output)
Initialise Score as string = Scorelbl.text
Initialise Total as string = Totallbl.text
Printline(4, Score & ", " & Total)
Questionbox.text = ""
Answerbox.text = ""
Scorelbl.text = ""
Totallbl.text = ""
Me.hide
Menu.show
Fileclose(4)

```

-Questions section will be the main section of the program. A random number will be generated which will act as an ID of a question so the question with that ID will be shown. A user input will be required to generate another question, using a button. Also, another user input that will be required is the answer to the question. Once the user inputs the answer, it will be compared to the actual answer stored in a csv file. The answers will be matched to the questions using the same ID, but in a different csv file. If the answer is correct, the user will get 1 point. If the question was answered incorrectly however, the score will stay the same, so "0/1" would be displayed and the correct answer will be shown in the answer box.

#### Progress

```

Fileopen(4, Location/Score.csv, openmode.output)
Initialise Fullline as string = Lineinput(4)
Initialise Parts() as string = fullline.split(",")
Scoretable.text = ("Score" & " " & "Out of")

```

Do

```
Scoretable.text = (Parts(0) & " " & part(1))
```

Loop until EOF

#### Usability Features

The navigation system around the program will be very easy to use due to its' simplicity. Once the user logs in, multiple buttons show up, which lead to different sections of the program. All buttons are labelled with the different sections of the program. In each section, there will be a button at the left lower corner which allows the user to get back to the main menu. The simplicity of the navigation system increases the efficiency when the program is used and makes the program more user-friendly.

In the questions section, the questions should be generated swiftly, without a delay. The answers should also be checked & the score recorded without any delays or errors. This will help to make the program more efficient as well.

Errors should not occur, especially if it's not caused by user input. Therefore, I have to make sure that all declared variables in my algorithms are in correct format, so that the user will not be able to enter letters or any other characters into an integer only variable.

The final solution of the program should be effective at teaching the user geography. To achieve this, when the user answers a question incorrectly, the right answer will be displayed so that the user can learn it. Also, progress of the user as they improve can be seen in the progress section of the program, which would determine the effectiveness of the program over time.

The visual design of the program should be not too bright or too dark so that the user can focus on its main purpose for longer without eye strain. The writing should also be clear, so that the writing stands out from the background and any images or tables.

#### Development

(Different versions of the program can be found in the "Development" folder.

##### Main

I have made multiple changes to the program over time so that it is more helpful to the user and it is more user-friendly. Unnecessary sections of the program were removed to avoid confusion and most of the processes that occur in the program are automatic, minimising the necessary user input. Also, multiple changes were made to minimise possible errors, bugs and crashes. Some improvements that had been made, improved the program in order to meet the user's requirements.

At first, when user registered, the username and the password was stored in a text file without a structure.



```

Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles RegisterConfirm
    Dim csvFile As String = My.Application.Info.DirectoryPath & "\login.csv"
    Dim outFile As IO.StreamWriter = My.Computer.FileSystem.OpenTextFileWriter(csvFile, False)

    outFile.WriteLine(username1.Text)
    outFile.WriteLine(password1.Text)
    outFile.Close()

    MsgBox("Registration Complete!", MsgBoxStyle.OkOnly)
    Me.Close()
    Start1.Show()
End Sub

```

This has been changed as this method caused many problems. For example, only one user could register as the previous username and password would be overwritten by the new. Also, This method meant that if there were more users' login information in the text file, then all accounts could be easily accessed by knowing one password as I would have to use ".contains" command which would search the text file for any password that the user typed in, accepting it as correct for any username.

Therefore, due to limitations this code has been changed so that unauthorised access is less likely as the person trying to log into another user's account would need the correct username and password of that account. The correct password must be matched with the username of the user now:

```

Sub registrationToFile()
    'Saves registration info into a csv file
    FileOpen(4, (My.Application.Info.DirectoryPath & "\login.csv"), OpenMode.Append)
    Dim fullline As String = username1.Text & "," & password1.Text

    PrintLine(4, fullline)

    FileClose(4)
End Sub

```

When the user registers, the information is stored in a csv file and the password is placed next to the username which means that during the login process, the password can be compared to the username and any other passwords will be incorrect during the login process. Also, this change allows multiple users to register, without overwriting the previous user. This means that user inputs can be stored separately now as well.

The old login process was made for 1 user only as the login information was stored in a text file and therefore, during registration process, the contents of that text file would be overwritten.

```

Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1_Click
    FileClose(1)

    Dim Logged As String() = System.IO.File.ReadAllLines(My.Application.Info.DirectoryPath & "\login.csv")
    Dim record = First Line In Logged

    If Logged(0) = LoginUsername.Text And Logged(1) = LoginPassword.Text Then
        MsgBox("Login Successful")
        Me.Hide()
        Menu1.Show()
        FileClose(1)
        LoginUsername.Text = ""
        LoginPassword.Text = ""
    Else
        Dim retry = MsgBox("Incorrect!", MsgBoxStyle.RetryCancel)
        If retry = Windows.Forms.DialogResult.Cancel Then
            FileClose(1)
            Me.Close()
        Else
            End If
    End If
End Sub

```

An alternative option was to search the text file line by line, by using a loop. However, without the record structure, the password and the username would have to be searched separately and the possibility of matching one user's username with another user's password from the file would still be

an issue. Also the issue of overwriting the previous user would still be a problem during the registration process so the loop would never be used. Therefore the login code has been changed to:

```
FileOpen(1, (My.Application.Info.DirectoryPath & "login.csv"), OpenMode.Input)
Dim counter As Integer
'loads the file into a record structure
Do Until EOF(1)
    counter = counter + 1
    Dim FullLine As String = LineInput(1)
    Dim Item() As String = Split(FullLine, ",")
    record(counter).username = Item(0)
    record(counter).password = Item(1)

    'Checks if the login and password are correct:
    If LogInUsername.Text = (record(counter).username) And LogInPassword.Text = (record(counter).password) Then
        'Log In
        Menu1.Show()
    End If
Loop
FileClose(1)
System.Threading.Thread.Sleep(1000)
DisconnectAll.Visible = True
Menu1.Show()
Menu1.Show()
```

Firstly, login information is now stored in a csv file, so that more than one accounts can be stored. Also, the record structure made it possible to accurately find the correct username and password when needed.

As shown in the image above, when the user tries to log in, the csv file which holds the login information is loaded into a structure. The user's login & password is searched line by line until the matching username is found. If the username or password was not found, it would mean that the user input does not exist in the file and therefore a message would display letting the user know that the login information that was entered is incorrect. Also, if the entered username is not on the same line as the password, the user will not log in as the username has to match the password correctly now.

At first, when the user answered a question, another fully random question would be generated from the csv file.

```
Randomize()
Number = (Int(Int(20 * Rnd()) + 1))
FileClose(2)
FileClose(1)
FileOpen(1, (My.Application.Info.DirectoryPath & "questions.csv"), OpenMode.Input)
Dim counter As Integer
Do Until EOF(1)
    counter = counter + 1
    Dim FullLine As String = LineInput(1)
    Dim Item() As String = Split(FullLine, ",")
    record(counter).Question = Item(0)
    record(counter).Answer = Item(1)
    record(counter).Level = Item(2)
Loop
NextQ.Visible = False
Submit.Visible = True
QuestionBox.Text = record(Number).Question ' Output Question from csv to the questionbox
FileOpen(2, (My.Application.Info.DirectoryPath & "TemporaryQ.csv"), OpenMode.Output)
PrintLine(2, record(Number).ID) 'Stores Current Question ID
FileClose(2)
FileClose(1)
```

However, if the user gets better over time and answers the questions correctly, harder questions would be displayed until the user got a question wrong. Therefore, a loop was added to search for questions with a higher difficulty level when a question was answered correctly. This has been done so that it helps the user to improve in this topic even more, especially with the difficult parts of geography. This also required to make a new field in the record structure: level. Each question was

assigned a difficulty from 1 to 3, 1 being the easiest and 3 being the hardest. If the user answers a question correctly, the level increases by 1 and if the question is answered incorrectly, the level decreases by 1.

```

10: def __init__(self, num_filters: int, kernel_size: int, stride: int, padding: int, bias: bool):
11:     self.num_filters = num_filters
12:     self.kernel_size = kernel_size
13:     self.stride = stride
14:     self.padding = padding
15:     self.bias = bias
16:
17:     # Create the kernel and bias
18:     self.kernel = self._create_kernel()
19:     self.bias = self._create_bias()
20:
21:     # Create the output
22:     self.output = self._compute_output()
23:
24:     # Return the output
25:     return self.output
26:
27: def _create_kernel(self):
28:     # Create the kernel
29:     kernel = np.zeros((self.kernel_size, self.kernel_size, self.num_filters))
30:
31:     # Fill the kernel with random values
32:     np.random.seed(0)
33:     kernel = np.random.randn(self.kernel_size, self.kernel_size, self.num_filters)
34:
35:     # Return the kernel
36:     return kernel
37:
38: def _create_bias(self):
39:     # Create the bias
40:     bias = np.zeros((self.num_filters))
41:
42:     # Fill the bias with random values
43:     np.random.seed(0)
44:     bias = np.random.randn(self.num_filters)
45:
46:     # Return the bias
47:     return bias
48:
49: def _compute_output(self):
50:     # Compute the output
51:     output = np.zeros((self.output_size, self.output_size, self.num_filters))
52:
53:     # Fill the output with random values
54:     np.random.seed(0)
55:     output = np.random.randn(self.output_size, self.output_size, self.num_filters)
56:
57:     # Return the output
58:     return output

```

This is the previous code of the “submit” button:

[illegible]

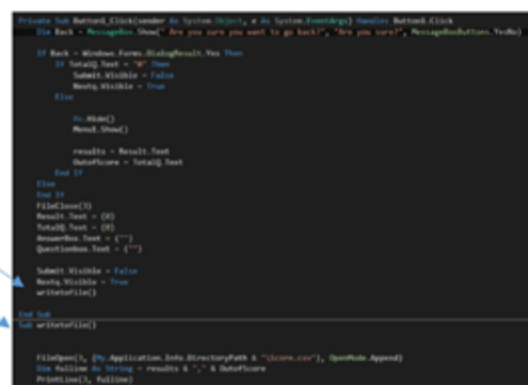
It has been changed to:

[illegible]

The submit button checks the user's answer with the correct answer in the csv file. The change that was made was that the decision is made whether to increase the difficulty for the user or lower it. If the user gets an answer correct then difficulty increases... Also, I have added ".toupper" to user's answer so that if the user enters an answer with a lower case first letter, it will not be affected the result anymore. I have also chosen to leave 2 buttons: "Submit" and "Next Question" instead of using one. I have done this so that when the user submits an incorrect answer, the correct answer will appear so that it helps the user to learn it and therefore the user can press the "Next Question" button when they are ready. Furthermore, when one button is usable, the other isn't to avoid errors and so that the user is unable to skip the question they don't know the answer to.

The "Back button" in the "Questions" section is very important in my program as it saves the final results of the user and stores it in a csv file. It's second minor purpose is to reset everything in the "Questions" section without having to restart the program.

At first, all results have been stored in one csv file:



```

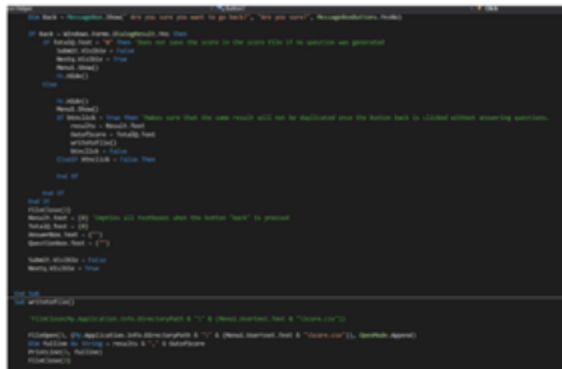
private void Submit_Click(object sender, EventArgs e) {
    // Back - Messagebox.Show("Are you sure you want to go back?", "Are you sure?", MessageBoxButtons.YesNo)
    // Back - Window.Focus(RichTextBox1)
    // Submit - Text = "Q"
    Submit.Visible = false
    NextQ.Visible = true
    else
    {
        // Show()
        MessageBox.Show()
        results = Result.Text
        Difficulty = TotalQ.Text
        if (TotalQ.Text == "0")
        {
            TotalQ.Text = "0"
            Results.Text = "0"
            AnswerBox.Text = ""
            QuestionBox.Text = ""
        }
        Submit.Visible = false
        NextQ.Visible = true
        WriteToFile()
    }
}

// Back
// WriteToFile()

FileOpen([Path.Application.Info.DirectoryPath & "Score.csv"], FileMode.Append)
// WriteToFile()
// WriteToFile()
Print(out(), FullLine)

```

However, this design meant that all results of all users would have been mixed up in one csv file which meant that if the user wanted to display the results, all would display. One way to deal with this problem was to make a new field in the csv file with a unique id of the user, which would be placed next to all the user's results so that when the user displays the results, only the results would be displayed with the user's unique id next to them. This would have to be done by making a search algorithm. However, I decided to create a separate file when the user successfully registers. After the user registers, a new csv file is created in the file that has been previously created, separating all results as they are saved in different files and in separate csv files.



The improved version firstly checks if a directory exists already, if it does then the new file and the csv files are not created so that they will not be overwritten. If it doesn't exist however, the new file is created with the csv file inside. The results are now saved into the new csv file of the user. This has been achieved by creating the directory with the user's username and therefore, the files will always have a unique name.

If the user wants to see their results, they can be displayed in the results table in the “Results” section.

```
Private Sub Button_Click(sender As System.Object, e As System.EventArgs) Handles UpdateBtn.Click
    FileOpen(1, My.Application.Info.DirectoryPath & "\score.csv", OpenMode.Input)
    LineInput(1)
    Dim counter As Integer
    counter = 0

    Do Until EOF(1)
        counter = counter + 1

        Dim FullLine As String = LineInput(1)

        Dim Line() As String = Split(FullLine, ",")
        record(counter).Surname = Line(0)
        record(counter).Total = Line(1)
        ScoreList.Items.Add(" " & record(counter).Surname & " " & record(counter).Total)
    Loop
    UpdateBtn.Visible = False
End Sub
```

At first, when the results were obtained from a single csv file, the csv was loaded into the list line by line, ready to display. However, this had to be slightly changed so that the results of a specific user were shown.

```
Private Sub Button_Click(sender As System.Object, e As System.EventArgs) Handles UpdateBtn.Click
    If System.IO.Directory.Exists(My.Application.Info.DirectoryPath & "/" & (Panel.Shortcut.Text) & "(Score.csv)" = False Then
        FileCreate()
        FileOpen(I, My.Application.Info.DirectoryPath & "/" & (Panel.Shortcut.Text) & "(Score.csv)", OpenMode.Create)
        LineInput(I)
        Dim counter As Integer = 0

        'Insert the file name
        Do Until EOF(I)
            counter = counter + 1
            Dim FullLine As String = LineInput(I)

            Dim Item() As String = Split(FullLine, ",")
            Record(counter).OverScore = Item(0)
            Record(counter).Total = Item(1)
            ScoreList.Item.Add(" " & Record(counter).OverScore & " " & Record(counter).Total)

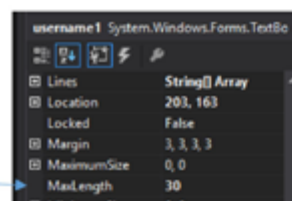
            'File is loaded into the listbox
        Loop
        UpdateBtn.Visible = False 'Prevents the display of duplicated results
    Else
        FileClose(I)

        MsgBox("No Results")
        UpdateBtn.Visible = False
    End If
End Sub
```

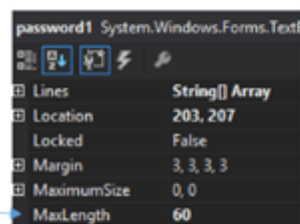
First, the code checks if the directory of the user exists to avoid a crash, if it does, the results of the user currently logged on are shown, by adding (Menu1.UserText) to the directory, making each user's file unique. This has solved the issue of displaying all users' results.

### User Inputs

Username: Can be letters or numbers. Username has a limit of 30 so that errors will not be caused by a very long username as it is used for file directory.



Password: Can be letters or numbers. Password also has a limit of 60 as it's not used for any other purpose than to login. However, it's unlikely that a user would have a password longer than 60 spaces.



Answers to the questions: Answers inputted by the users has a high limit as the size of the answer will not affect anything. Any characters can be used. User input is compared to the correct answer in a csv file, determining if it's correct or not. It will not matter if the user input is in lower or upper case as the program converts the answer to upper case letters.

```
If AnswerBox.Text.ToUpper = record(Number).Answer.ToUpper Then
```

After the previous change, I also changed it so that if there are any spaces, they will be deleted as the space would cause the answer to be wrong.

```
If AnswerBox.Text.ToUpper.Trim = record(Number).Answer.ToUpper.Trim Then
    MsgBox("Correct")
    TotalQ.Text = TotalQ.Text + 1
    Result.Text = Result.Text + 1
    check = True
```

**Comment [JM3]:** Review from stakeholder at each stage

### Testing

Navigation (buttons)

Start-up window:



Register button:



2<sup>nd</sup> Register button:





Quit button:



2<sup>nd</sup> Quit button:



Login Button:



## Examiner commentary

### Question/Part: AO 2.2 Analysis

#### Marks; 5/10

Introduction to the project is detailed and clear. Computational methods are attempted, with some correct uses and applications and the student has tried to relate concepts to their own project. There is some detail about the stakeholders although this is quite vague and more depth could be added.

The research into existing systems is very limited, although does identify a few features to take forward, however there is a lack of evidence as to where the majority of the identified user requirements come from.

A set of basic requirements are described but some lack explanation. There is some relation to what evidence may be used to measure success, however this is brief.

A simple limitation is described, alongside two Visual Basic issues which do not seem relevant.

The real issue with this section is the lack of real investigation, however although possibly slightly generous the teacher's mark can be agreed as best-fit.

### Question/Part: AO 3.1 Design

#### Marks; 5/15

The teacher's mark is fair for the Design section, which is lacking real detail, does not cover all of the descriptor points in mark band 2.

The GUI designs are explained in terms of functionality and usability issues covered in several places throughout the section. The description of each screen does show some awareness of how the solution can be decomposed and the following sections describe some of the aspects that will form the program.

A diagram is used to represent a simple outline of the system.

There is an attempt to show some of the algorithms that may be used to create a solution. The simpler log in and menu pseudocode are both usable, however the questions code can only be followed when reading the description underneath. The pseudocode is also poorly presented with no indentation to make it readable.

Data structures and file use is outlined but there is no detail as to actual database fields or file structures. The use of arrays and variables is described but there is no accompanying data dictionary where any key variables are identified and no data type or validation detail is explained.

There are descriptions of a number of approaches to testing. Some aspects of the system lack specific examples of how they may be tested or with what data. There is no description of post-development testing.

### Question/Part: AO 3.2 Developing the coded solution

#### Marks; 7/15

The teacher's mark is generous based on the evidence provided in this section. It is unclear as to whether the implementation was documented during development. The past tense used to describe the work carried out seems to confirm that documentation was written post-development.

The student has attempted to describe some parts of the development process and changes made to their original idea. There are a number of screenshots used, however it is not always clear what is being shown or discussed in relation to them and some are very small, which makes it hard to see what is being described.

The code provided shows a simple but modular event-driven approach. There are few comments to identify some modules of code. Most variables and subroutines are appropriately named.

### Question/Part: AO 3.2 Testing to inform development

#### Marks; 4/10

This section has been marked generously.

There is a lack of evidence that any testing was carried out during development. Errors are explained and some before and after pieces of code are provided in the development section.

There are no screenshots of the code running or interacting with the GUI at these points so the tests mentioned can be seen working or errors being thrown. Descriptions of errors and tests should be reinforced with real evidence in the form of screenshots or video.

The screenshots in the Testing section have been identified by the teacher as functionality tests. Marks should not be awarded for functionality tests in this section.

### Question/Part: AO 3.3 Testing to inform evaluation

#### Marks; 2/5

The Testing section appears to have been created post-development. If this work was intended to show developmental testing it would be more appropriate to see the screenshots in the relevant sections of development to show how the modules of code relating to those parts were tested during the iterations.

There is an attempt to describe and evidence some robustness testing but there is not enough evidence of end user feedback or comment on usability to award more marks than those given.

### Question/Part: AO 3.3 Evaluation of solution

#### Marks; 4/15

There is some discussion of how well some of the requirements identified have been met, however the fact that several of the initial requirements from the Analysis section have not been attempted is not really addressed.

The teacher's comments and marking are accurate for this section. It is not well written, not entirely realistic and it does not cross reference any evidence in the project to back-up any of the discussion.



We'd like to know your view on the resources we produce. By clicking on the 'Like' or 'Dislike' button you can help us to ensure that our resources work for you. When the email template pops up please add additional comments if you wish and then just click 'Send'. Thank you.

Whether you already offer OCR qualifications, are new to OCR, or are considering switching from your current provider/awarding organisation, you can request more information by completing the Expression of Interest form which can be found here:

[www.ocr.org.uk/expression-of-interest](http://www.ocr.org.uk/expression-of-interest)

#### OCR Resources: *the small print*

OCR's resources are provided to support the delivery of OCR qualifications, but in no way constitute an endorsed teaching method that is required by OCR. Whilst every effort is made to ensure the accuracy of the content, OCR cannot be held responsible for any errors or omissions within these resources. We update our resources on a regular basis, so please check the OCR website to ensure you have the most up to date version.

This resource may be freely copied and distributed, as long as the OCR logo and this small print remain intact and OCR is acknowledged as the originator of this work.

OCR acknowledges the use of the following content:  
Square down and Square up: alexwhite/Shutterstock.com

Please get in touch if you want to discuss the accessibility of resources we offer to support delivery of our qualifications:  
[resources.feedback@ocr.org.uk](mailto:resources.feedback@ocr.org.uk)

#### Looking for a resource?

There is now a quick and easy search tool to help find **free** resources for your qualification:

[www.ocr.org.uk/i-want-to/find-resources/](http://www.ocr.org.uk/i-want-to/find-resources/)

[www.ocr.org.uk/alevelreform](http://www.ocr.org.uk/alevelreform)

#### OCR Customer Contact Centre

##### General qualifications

Telephone 01223 553998

Facsimile 01223 552627

Email [general.qualifications@ocr.org.uk](mailto:general.qualifications@ocr.org.uk)

OCR is part of Cambridge Assessment, a department of the University of Cambridge. *For staff training purposes and as part of our quality assurance programme your call may be recorded or monitored.*

© **OCR 2017** Oxford Cambridge and RSA Examinations is a Company Limited by Guarantee. Registered in England. Registered office 1 Hills Road, Cambridge CB1 2EU. Registered company number 3484466. OCR is an exempt charity.

